Blessed are all who spread His Glory.

BE INFORMED, The "**xlsx.zip**" is a 'zip file' with all scripture in .xlsx format.
BE INFORMED, The "**Alfabet.zip**" is a 'zip file' with all scripture letters in pic and audio format to use.

I am a loner and recluse, but have made a code that I will paste here and you can test the code as I will repeat many times, please test all code first in an AI (if you want to decipher the code for maliciousness) and still only run the code in a Windows 10 or 11 Sandbox or Virtual Machine (VM) for extra safety.

I want to share my code with you. In the Holy name of the Blood of:

Jee-sh-oeaijn	**ישו ע**	י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ש	Sh	Eat, Consume, Destroy, A rock	Teeth
		ו	Oe	Join, Making secure	Nail
		ע	Eij	See, Know, Watch, Twist, Dark, Experience	Eye

(The) Arm(s) (are) Destroy(ed) (by) Nail(s) See!

I will only in PDF visually code, so you can see and I hope you will want to see. It will be, "**EXCEL naar csv**" (48 kB) and the second code "**Letterlijk**" is (complete) ~37 MB **"Alfabet.zip"** with pics and audio, therefore I give you the 13 kB code only in text, and **"xlsx.zip"** and **"Alfabet.zip"** as optional downloads.

See if you trust me. First test the code in Copilot AI, or other web based AI without login. I have split the code visually with marker text, so you can copy past it easier and check the parts and when completed the whole code. Just ask what the code does and if it is malicious, virus, trojan or not. The contents of the 2 **".bat"** files can be copied and pasted in one go, per **".bat"** file.

For my program needs scripture (you can copy from e-sword all text into Excel and sanitize the text in my program or make a .csv). If you don't trust me or do not care/want this. Ignore or block me and I will not contact you ever again, for I respect your choice. Most likely I won't after this though.

My Code needs "PowerShell 7" and the programs need scripture in .xlsx files to convert and sanitize (remove diacritical's). Or .csv files with only the sentences in Hebrew, so no chapter, book or verse, line by line.

The code needs a preset "Folder structure" to function even the empty folders have a function as you will read later, the folders will be made in the "**.bat**" file if not present on the desktop.

The whole Hebrew Elephvet (Alphabet (In my version Aflabet = Dutch)), the .png or .jpg as well as a short and -long audio file (.mp3), are not included for bloating prevention, but I hope you see how I meant it to work and you can make it even better than I ever could.

All **.xlsx** files, and therefore all generated **.csv** files, are "sanitized" (without diacritical marks) Hebrew text and letters of all of Scripture and all modern letters (for the new testament version like Brackets and numbers) removed. Yes even the New testament in Hebrew, is as I could copy from the program E-Sword in Hebrew, you can too and in **.xlsx** file format my program can remove all diacritical's, Or

you have to do it in **.csv** manually, sorry. Forgot to program it in there and (If true) remaining time might be very short.

You are free to change any and all of the code to use it with Greek or other languages.

I hope you will test my program on an VM, because the code is open, you can copy paste everything into an AI to check the code on "malicious" code before running as described before. I have none in it, but "man in the middle attacks" could corrupt my files.

Best and safest way is to run the code is in a Windows Sandbox on Windows 10 or 11. No malformed code, virus or malware can do harm there, please accept in Sandbox for the 5 minutes to download the PowerShell 7 files needed to run the code. All files needs to be on Desktop.

The code does not work on PowerShell 5.1 because of the used code components.

Manual, for Windows 10/11 (or Sandbox), to install PowerShell 7:

1) **Program "EXCEL naar csv":**
   Comment: "naar" is Dutch for "to", so "EXCEL naar csv.ps1" is just "EXCEL to csv.ps1"

   To start "EXCEL naar csv.ps1" in PowerShell 7.
   - Start **EXCEL naar csv.bat** ← Looks if PowerShell 7 is installed if not downloads and installs it, then starts the program.

2) **Program "Letterlijk":**
   Comment: "Letterlijk" is Dutch for "Literally", so "Letterlijk.ps1" is just "Literally.ps1".

   To start "Letterlijk.ps1" in PowerShell 7.
   - Start **Letterlijk.bat** ← Looks if PowerShell 7 is installed if not downloads and installs it, then starts the program.

Without the pics and audio the program does not work correctly, as in it has no pictures to display nor audio to play, but you can change that before running the code as I will describe later. The Code itself is just a 13 kB script with a 1kB ".bat" (starter file).

Ok, wat do I claim the code will do and why did I send you?

Program "EXCEL naar csv" on start you get two options**:**
Option 1:
Choose to convert all .xlsx files into larger .csv (comma separated variables) files, which are easier to decode in my program. Or you can just convert one file or a few (one at a time) need to restart each time though.

That is all it does. Nothing more, nothing less.

Once you have the .csv file(s) and choose option 2, on restart of the program, you can do a lot of Scripture Decoding.

Option 2 functions:
1) **Search word:** Use Hebrew words and letters, what word are you looking for?

2) **Max distance letters**: How far can each letter be from one and other to form the word with a set distance e.g. all letters at 1000 letters apart? (When max the program calculates the length of the word and maximum spacing but can take a long time to finish and fully utilize the CPU! Watch out. You can also set the number of Threads assigned before searching.

3) **Remove Spaces Checkbox:** Gives you a continuous letter combination as a Clean_[FILENAME].csv file. This so you can search "The Bible code" kind of search, without spaces, Maqaf's, newlines and carry return counting as a letter.

4) **HTML block length:** Set the length of the letters per row / line for the HTML Filter, can only be used when Remove Spaces is active, or it will stay as normal line by line from scripture.

5) **HTML Filter:** Gives you an overview of the whole scripture with the matching words. Yellow red for found words. Red White letters for the "Selected" "**Distance**" between letters.

6) **Threads:** To set the CPU count. When using all your system Threads your system will use 100% and will be "locked out" till completion. So be careful. Or be sure you won't be using you PC during search with all Threads.

7) **Analysis Checkbox:** Once created it has a short log (is the "Log" button for and only usable after running a search) to see the "found count" and search time.

8) **Graph**: Gives you a total view of all found words in the text after a search. With mouse click, hold and drag you can zoom in, "double click" or "button" tot reset zoom. Can be made a ".png, .jpg, or even .pdf" but not sure if it is useful.

9) **Export**: Only for other path's. It does "export" the Clean_ automatically, when selected with "**Remove Spaces Checkbox"** and "**HTML block length**".

Program "Letterlijk":
This is the (complete version with pic and audio) ~38MB file. So what is it?

I think you know the first word of Genisis:

In the beginning	בראשית	ב	Ve	In, Family, House, The body, Whitin	House
		ר	We	Captain, Authority, Chief, Person, The highest	Man Head
		א	Eh	Adanay, Strong, Power, Beginning, Leader, 1st	The Creator
		ש	Sh	Eat, Consume, Destroy, A rock	Teeth
		י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ת	Th	Last, Mark, Sign, Seal, Ownership, Covenant	Cross

Family (of) Man(kind) (The) Creator Destroy(s) (his) Arm(s) (on the) Cross.

This program needs the .csv file not as one big block. But as words with Maqaf's, so not the "Clean_*.csv" version (all text as continuous text) but the original "*.csv" file.

It reads the loaded scripture and reads it letter for letter, With a depiction (image) of the letter, with either the short or the -long audio version.

You can use your voice and create the audio, as long as you use the format "א.mp3" as e.g. "El, Eleph or Aleph, Depicted as The Creator" and "א-long.mp3" as e.g. "El, Eleph or Aleph, Depicted as The Creator. Meanings: Adonai. Strong. Power. Beginning. Leader. 1st." and for picture format "א.png" or "א.jpg", etc. for every Hebrew letter of the Elephvet / Alfabet. (Alfabet = Dutch for Alphabet) I wrote in my version the same as the audio with a nice Background. But I am sure you can make even better ones.

I Used e.g. Mem, I had an mountain and a single bright heavenly light beam breaking through the clouds with a waterfall and small river (representing water) in the distant back with lightning and a raging storm around flowing into the wild ocean / sea in the foreground (representing chaos) with one boat with the word "Water" and the second "Chaos".

I wanted to include my version of the pic here, but you might see it as an "advertisement or malicious pic" so I have not. But I hope you can "visualize" my pic and make better ones for yourself and/or your teachings.

The pic files and the audio have to be in the same folder of the csv files with one deeper folder called Folder "Alfabet".

The folder structure needs to be:
The word will be displayed without any files, but no image will be displayed and no sound will be played, so no explanation like in the examples, just the letters as word.

Wherever the .ps1 files are there has to be the Folder "CSV" and the Nested Folder "Alfabet" and in the folder "Alfabet" place all the alphabet letters as ".png or .jpg" and audio files as ".mp3's". Done.

The folders will be made in my "**.bat**" if they are not there.

Folder structure:
.ps1 files
--- xlsx (Folder for xlsx files optional, always set browsable if unavailable) ---
CSV (folder)
|
--- Alfabet (Folder)
|
--- All the alphabet letters as ".png or .jpg" and audio files as ".mp3" or "-long.mp3".

Let a Hebrew explain the words they seem to know, but see past the literal letters of that same word. And explain it. My program can translate the letter of scripture word by word. Please show the Jew, the Muslim and all who want or need to see the literal letter of the Old Testament, just like the example shown above and examples below.

You can make your own .csv file with one or many sentences or words and use my "**Letterlijk**" program to show the letters and their meaning, with audio in short or long format this will clear many things I hope.

Why I made the program is also because of the claim "we don't know" the meaning of:

It is clear and unambiguous for me.

with	את	א	Adanay, Strong, Power, Beginning, Leader, 1st	The Creator
		ת	Last, Mark, Sign, Seal, Ownership, Covenant	Cross/Own

את = (The) Creator (on the) Cross.

and with	ואת	ו	Join, Making secure	Nail/Pierce
		א	Adanay, Strong, Power, Beginning, Leader, 1st	The Creator
		ת	Last, Mark, Sign, Seal, Ownership, Covenant	Cross/Own

ואת = Nail(ed) (the) Creator (on the) Cross.

Place all of my code as files with correct file names (or adjust in the code to your naming) on desktop (with folders) and all code will run correct or adjust the code so it runs from the correct path, you can ask In Sandbox Copilot or other “AI”LLM” to do that for you, if you cannot.

"!pwshExe!" -ExecutionPolicy Bypass -File "%USERPROFILE%\Desktop\Letterlijk.ps1"

Paths look like: "%USERPROFILE%\Desktop\Letterlijk.ps1" ← Predefined location.

Paths look like: $fileDialog.InitialDirectory = Join-Path $PSScriptRoot "xlsx" ← wherever you start the script.

Again to emphasize: Please use a VM (Virtual Machine) or Windows Sandbox so nothing can ever go wrong.

That way you can see how I “simply” wanted it to work. You can use any Hebrew sentence in a .csv file to display every letter, or copy paste whole scripture and send .xlsx files after converting it into .csv (those are 2x as big that’s why I send the .xlsx files) and you do not need Office for those to be converted.

On final thought. A staff, turned around with a point (or nail) becomes a spear. That in mind you will see why I added “Staff or Spear(ed)” for it makes everything much clearer when you read it with that “addition” witch still holds true to the meaning and value of the letter.

A few more examples:

Jeehoehe	יהוה	י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ה	He	Look, Show, Reveal, Window, Breath	Praise or Reveal
		ו	Oe	Join, Making secure	Nail
		ה	He	Look, Show, Reveal, Window, Breath	Praise or Reveal
Aijmnoeel	עמנואל	ע	Eij	See, Know, Watch, Twist, Dark, Experience	Eye
		מ	Me	Chaos, Water, Mighty, Massive	Water
		נ	Ne	Son, Heir, Grow, Life, Action	Seed or Spread
		ו	Oe	Join, Making secure	Nail
		א	Eh	Adanay, Strong, Power, Beginning, Leader, 1st	The Creator
		ל	Le	Instruct, Teach, Reward, From, To	Staff or Speer(ed)
Vth / Beth	בית	ב	Ve	In, Family, House, The body, Whitin	House
		י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ת	Th	Last, Mark, Sign, Seal, Ownership, Covenant	Cross
Lchm / Lechem	לחם	ל	Le	Instruct, Teach, Reward, From, To	Staff or Speer(ed)
		ח	Ge	Divide, Seperate, Protect, Fence	Wall
		ם	Me_	Chaos, Water, Mighty, Massive	Water
Jshwel / Israel	ישרא ל	י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ש	Sh	Eat, Consume, Destroy, A rock	Teeth
		ר	We	Captain, Authority, Chief, Person, The highest	Man Head
		א	Eh	Adanay, Strong, Power, Beginning, Leader, 1st	The Creator
		ל	Le	Instruct, Teach, Reward, From, To	Staff or Speer(ed)
Jwoeshlm / Jerusalem	ירושל ם	י	Je	He, My, Strength, Worship, Work, Throw, Them	Arm
		ר	We	Captain, Authority, Chief, Person, The highest	Man Head
		ו	Oe	Join, Making secure	Nail
		ש	Sh	Eat, Consume, Destroy, A rock	Teeth
		ל	Le	Instruct, Teach, Reward, From, To	Staff or Speer(ed)
		ם	Me_	Chaos, Water, Mighty, Massive	Water

I use this table:

א	Adonai, Strong, Power, Beginning, Leader, 1st	The Creator
ב	In, Family, House, The body, Within	House
ג	Walk, Gather, Pride, With, But	Foot
ד	Entrance, Path, Hang, A way of life	Door
ה	Look, Show, Reveal, Window, Breath	Praise or Reveal
ו	Join, Making secure, Fasten	Nail
ז	Pierce, Sword, Plow, Cut, Ax	Weapon or Tool
ח	Divide, Seperate, Protect, Fence	Wall
ט	Contain, Surround, Twist,Mud	Basket
י	He, My, Strength, Worship, Work, Throw, Them	Arm
כ	Tame, Bend, Open, Allow, A wing, To cover	Hand palm
ל	Instruct, Teach, Reward, From, To	Staff or Speer(ed)
מ	Chaos, Water, Mighty, Massive	Water
נ	Son, Heir, Grow, Life, Action	Seed or Spread
ס	Aid, Support, Assist, Protect, Sharp	Shield or Help
ע	See, Know, Watch, Twist, Dark, Experience	Eye
פ	Opening, Speak, Command, Edge, Blow, Scatter	Mouth
צ	Journey, Desire, Need, Hunt, Chase, Pull toward	Side (of man)
ק	Circle, Sun, Time, Behind, Last, Final, The least	Sun or Final
ר	Captain, Authority, Chief, Person, The highest	Man Head
ש	Eat, Consume, Destroy, A rock	Teeth or Destroy
ת	Last, Mark, Sign, Seal, Ownership, Covenant	Cross
ך	Tame, Bend, Open, Allow, A wing, To cover	Hand palm
ם	Chaos, Water, Mighty, Massive	Water
ן	Son, Heir, Grow, Life, Action	Seed
ף	Opening, Speak, Command, Edge, Blow, Scatter	Mouth
ץ	Journey, Desire, Need, Hunt, Chase, Pull toward	Side (of man)

Sincerely and hopefully see you in heaven, **Concerned_Man**

So here is de plain text code the rest you can do manually if you choose to do the converting with e-sword to either the right .xlsx or .csv **without** diacritical's (just the Hebrew text, no chapters or books):

I have # ^^^^^ **Copy before for AI Check Checkblock 1** ^^^^^ so you will not exceed the 10240 letter limit if you want to check my code in an AI / LLM like Grok, Copilot, CHAT GPT, etc.. (I hope all have the same limit I used Copilot to check my code)

In **Excel naar csv.ps1** # ^^^^^ **Copy before for AI Check Checkblock 7** ^^^^^ In **Letterlijk.ps1** # ^^^^^ **Copy before for AI Check Checkblock 2** ^^^^^

Excel naar csv.bat: _____

```
@echo off
setlocal EnableDelayedExpansion

REM -- Detect Windows version -- for /f
"tokens=4-5 delims=. " %%i in ('ver') do (
   set "winVer=%%i.%%j"
)

REM -- Detect Sandbox -- set "isSandbox=false" echo
%USERPROFILE% | findstr /i "WDAGUtilityAccount" >nul if
!errorlevel! == 0 (
   set "isSandbox=true"
)

REM -- Define paths --
set "PS7Path=C:\Program Files\PowerShell\7" set
"PS7Exe=%PS7Path%\pwsh.exe" set
"DesktopPath=%USERPROFILE%\Desktop\PowerShell" set
"DesktopCSVPath=%USERPROFILE%\Desktop\CSV\Alfabet"
set "ZipPath=%DesktopPath%\PowerShell7"
set "zipFile=%DesktopPath%\PowerShell.zip"
set "msiFile=%TEMP%\pwsh7.msi" set
"pwshCmd="

if not exist "!DesktopPath!" (
mkdir "!DesktopPath!"
   mkdir "!DesktopCSVPath!"
)

REM -- Check if pwsh is available via PATH --
where pwsh >nul 2>nul if !errorlevel! == 0 (     echo
? PowerShell 7 is already available via PATH.
   set "pwshCmd=pwsh"
) else (
   if "!isSandbox!"=="true" (
      REM -- Check if pwsh.exe already exists in ZIP path --
if exist "!ZipPath!\pwsh.exe" (            echo ? PowerShell 7
already extracted in Sandbox.
         set "pwshCmd=!ZipPath!\pwsh.exe"
      ) else (
         echo ? Downloading PowerShell 7 ZIP for Sandbox...
powershell -Command "Invoke-WebRequest -Uri
'https://github.com/PowerShell/PowerShell/releases/download/v7.4.0/PowerShell-7.4.0-win-x64.zip'
OutFile '!zipFile!'"           powershell -Command "Expand-Archive -Path '!zipFile!' -DestinationPath
'!ZipPath!' -Force"            set "pwshCmd=!ZipPath!\pwsh.exe"
      )
```

```
    ) else (
        REM -- Host: Try WinGet first --         where
winget >nul 2>nul         if !errorlevel! == 0 (
echo ? Installing PowerShell 7 via WinGet...
winget install --id Microsoft.PowerShell --source
winget --accept-package-agreements -accept-
source-agreements
        ) else (
            echo ? WinGet not available. Downloading MSI installer...
            powershell -Command "Invoke-WebRequest -Uri
'https://github.com/PowerShell/PowerShell/releases/download/v7.4.0/PowerShell-7.4.0-win-x64.msi'
OutFile '!msiFile!'"
            msiexec /i "!msiFile!" /quiet /norestart
        )

        REM -- Verify installation --         if exist
"!PS7Exe!" (             echo ? PowerShell 7
installed successfully.
            set "pwshCmd=!PS7Exe!"

            REM -- Add to user PATH for Windows 10 and 11 --
echo ? Adding PowerShell 7 to user PATH...
            setx PATH "%PATH%;!PS7Path!" >nul
        ) else (             echo ? Installation failed.
Exiting script.
            goto :eof
        )
    )
)

REM -- Launch your PowerShell script -- echo
?? Launching EXCEL naar csv.ps1...
"!pwshCmd!" -ExecutionPolicy Bypass -File "%USERPROFILE%\Desktop\EXCEL naar csv.ps1"

Endlocal
```

Excel naar csv.bat: _____

Excel naar csv.ps1: _____

```
# ? Forceer adminrechten (werkt voor zowel PowerShell Core als Windows PowerShell) if
(-not ([Security.Principal.WindowsPrincipal] `
    [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole(`
    [Security.Principal.WindowsBuiltInRole]::Administrator)) {

    $exe = if ($PSVersionTable.PSEdition -eq 'Core') { 'pwsh.exe' } else { 'powershell.exe' }

    Start-Process $exe -Verb RunAs -ArgumentList @(
        "-NoProfile",
        "-ExecutionPolicy", "Bypass",
```

```
        "-File", "`"$PSCommandPath`""
    )
exit
}

# Initialisatie: Blok 1
Set-ExecutionPolicy Bypass -Scope Process
[System.Threading.Thread]::CurrentThread.ApartmentState = "STA"
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName Microsoft.VisualBasic
Add-Type -AssemblyName System.Windows.Forms.DataVisualization if
(-not (Get-Module -ListAvailable -Name ImportExcel)) {
    Install-Module -Name ImportExcel -Scope CurrentUser -Force
}
Import-Module ImportExcel

# Menu: keuze maken: Blok 2 function
Show-Menu {
    $global:csvPad = ""
    $global:csvFolder = ""
    $global:zoekterm = ""
    $global:results = @()
    $global:progressBar = New-Object System.Windows.Forms.ProgressBar
    $global:statusLabel = New-Object System.Windows.Forms.Label

    [Microsoft.VisualBasic.Interaction]::InputBox(
        "Choose action:`n1. Convert Excel to CSV`n2. Analyse CSV via GUI",
        "PowerShell CSV Tool",
        "2"
    )
}

# Update Result : Blok 3 function
Update-ResultBoxSafely {
    param (
        [System.Windows.Forms.TextBox]$box,
        [System.Collections.ArrayList]$results
    )

    $text = ($results | ForEach-Object {
        "Afstand $($_.Afstand) → Matches op: $($_.Posities -join ', ')"
}) -join "`r`n"

    [System.Windows.Forms.Application]::DoEvents()
    $box.Text = $text
    $box.Refresh()
}

# Clear Memory: Blok 4 function
Clear-mem {
```

```
    # NU pas: opruimen en reset
    $global:results = @()
    Remove-Variable -Name results -Scope Global -ErrorAction SilentlyContinue
            Remove-Variable -Name currentAfstand -Scope Global -ErrorAction SilentlyContinue
    [System.GC]::Collect(2, [System.GCCollectionMode]::Forced)
    [System.GC]::WaitForPendingFinalizers()
            $resultBox.Clear()
    $global:progressBar.Value = 1
    $global:statusLabel.Text = "Ready for new Analysis"
    [System.Windows.Forms.Application]::DoEvents()
}

# Conversie Excel naar CSV: Blok 5 function
Convert-ExcelToCsv {
    [CmdletBinding()]
    param ()

    # 1) Laad Forms voor dialoogvensters
    Add-Type -AssemblyName System.Windows.Forms

    # 2) Vraag: bulk of single?
    $result = [System.Windows.Forms.MessageBox]::Show(
        "Do you want to convert all .xlsx files in the folder you select?`r`n`r`nOn NO you can select a
single .xlsx file.",
        'Bulk conversion',
        [System.Windows.Forms.MessageBoxButtons]::YesNo,
        [System.Windows.Forms.MessageBoxIcon]::Question
    )

    $selectedPath =Join-Path $PSScriptRoot "CSV"
    if (-not (Test-Path $selectedPath)) {
        New-Item -ItemType Directory -Path $selectedPath | Out-Null
    }

    if ($result -eq [System.Windows.Forms.DialogResult]::Yes) {
        # 2a) Bulk: kies input-folder
        $inDialog = New-Object System.Windows.Forms.FolderBrowserDialog
        $inDialog.Description       = 'Select Folder with .xlsx files'
        $inDialog.ShowNewFolderButton = $false
if ($inDialog.ShowDialog() -ne 'OK') {
            [System.Windows.Forms.MessageBox]::Show(
                'Bulk conversion cancelled.',
                'Cancelled',
                [System.Windows.Forms.MessageBoxButtons]::OK,
                [System.Windows.Forms.MessageBoxIcon]::Information
            )
return
        }
        $inputFolder = $inDialog.SelectedPath
```

```
        # Alle .xlsx in die map
        $files = Get-ChildItem -Path $inputFolder -Filter *.xlsx -File
if ($files.Count -eq 0) {
            [System.Windows.Forms.MessageBox]::Show(
               'No .xlsx FIles found in the Folder.',
               'No Files',
               [System.Windows.Forms.MessageBoxButtons]::OK,
               [System.Windows.Forms.MessageBoxIcon]::Information
            )
return        }
    } else {
        # 2b) Single: kies 1 bestand
        $fileDialog = New-Object System.Windows.Forms.OpenFileDialog
        $fileDialog.InitialDirectory = Join-Path $PSScriptRoot "xlsx"
        $fileDialog.Filter           = 'Excel files (*.xlsx)|*.xlsx|All files (*.*)|*.*'
        $fileDialog.Title            = 'Select one .xlsx file'
if ($fileDialog.ShowDialog() -ne 'OK') {
            [System.Windows.Forms.MessageBox]::Show(
               'Single conversion cancelled.',
               'Cancelled',
               [System.Windows.Forms.MessageBoxButtons]::OK,
               [System.Windows.Forms.MessageBoxIcon]::Information
            )
return
        }
        $files = ,(Get-Item -LiteralPath $fileDialog.FileName)
    }

    # 3) Kies output-folder (voor zowel bulk als single)
    $outDialog = New-Object System.Windows.Forms.FolderBrowserDialog
    $outDialog.Description       = 'Select the folder for the .csv outputs'
    $outDialog.ShowNewFolderButton = $true
    $outDialog.SelectedPath  = Join-Path $PSScriptRoot "CSV\"
if ($outDialog.ShowDialog() -ne 'OK') {
        [System.Windows.Forms.MessageBox]::Show(
           'No output Folder selected.',
           'Cancelled',
           [System.Windows.Forms.MessageBoxButtons]::OK,
           [System.Windows.Forms.MessageBoxIcon]::Information
        )
return
    }
    $outputFolder =  $outDialog.SelectedPath

    try {
        # 4) Loop over elk .xlsx-bestand
foreach ($file in $files) {
            $path     = $file.FullName
            $baseName = [IO.Path]::GetFileNameWithoutExtension($path)
```

```
        # 4a) Bepaal sheets
        $sheets = (Get-ExcelSheetInfo -Path $path).Name

        foreach ($sheet in $sheets) {
          # 4b) Importeer data en maak CSV-regels
          $data     = Import-Excel -Path $path -WorksheetName $sheet
          $csvLines = $data | ConvertTo-Csv -NoTypeInformation

          # 4c) Strip enkel de rand-" per regel
$csvLines = $csvLines | ForEach-Object {                    if
($_.StartsWith("") -and $_.EndsWith("")) {
                $_.Substring(1, $_.Length - 2)
              }
else { $_ }
          }

          # 4d) Schrijf naar disk met UTF8-BOM
          # $csvFile  = "$baseName-$sheet.csv"
          $csvFile  = "$sheet.csv"
          $csvPath  = Join-Path $outputFolder $csvFile
          $bom      = [char]0xFEFF
          $content  = $bom + ($csvLines -join "`r`n")
          [IO.File]::WriteAllText($csvPath, $content, [Text.Encoding]::UTF8)

          Write-Host "Converted: $csvFile"
        }
      }

      [System.Windows.Forms.MessageBox]::Show(
        'Conversion Ready!',
        'Done',
        [System.Windows.Forms.MessageBoxButtons]::OK,
        [System.Windows.Forms.MessageBoxIcon]::Information
      )
}
  catch {
      [System.Windows.Forms.MessageBox]::Show(
        "ERROR during conversion:`n$($_.Exception.Message)",
        'Error',
        [System.Windows.Forms.MessageBoxButtons]::OK,
        [System.Windows.Forms.MessageBoxIcon]::Error
      )
  }
}

# ^^^^^ Copy before for AI Check  Checkblock 1 ^^^^^

# Analyse Multithread: Blok 6 function
Analyse-SprongMultithread {
  param (
```

```
        [char[]]$zoekterm,
        [char[]]$inhoud,
        [int]$inhoudLength,
        [int]$woordLength,
        [int]$maxAfstand,
        [int]$maxThreads
    )

    $global:results = [System.Collections.ArrayList]::new()

    # Scriptblock als string voor veilige overdracht
    $scriptBlock = {
        param ($afstand, $zoekterm, $inhoud, $inhoudLength, $woordLength)

        $global:currentAfstand = $afstand

        $maxStart = $inhoudLength - (($woordLength - 1) * $afstand) - 1
        $matches = New-Object 'System.Collections.Generic.List[int]'

        for ($start = 0; $start -le $maxStart; $start++) {
            $gevonden = $true             for ($i = 0; $i
-lt $woordLength; $i++) {                 $pos =
$start + ($i * $afstand)                if
($inhoud[$pos] -ne $zoekterm[$i]) {
                    $gevonden = $false
                    break
                }
            }
            if ($gevonden) { $matches.Add($start) }
        }

        return [PSCustomObject]@{
            Afstand = $afstand
            Posities = $matches
            IsMatch = ($matches.Count -gt 0)
        }
    }

    # RunspacePool opzetten
    $runspacePool = [runspacefactory]::CreateRunspacePool(1, $maxThreads)
    $runspacePool.Open()
    $powershells = @()
    $jobs = @()

    for ($afstand = 1; $afstand -le $maxAfstand; $afstand++) {
        $ps = [powershell]::Create()
        $ps.RunspacePool = $runspacePool
        $null = $ps.AddScript($scriptBlock)
        $null = $ps.AddArgument($afstand)
        $null = $ps.AddArgument($zoekterm)
```

```
        $null = $ps.AddArgument($inhoud)
        $null = $ps.AddArgument($inhoudLength)
        $null = $ps.AddArgument($woordLength)
        $powershells += $ps
        $jobs += $ps.BeginInvoke()
    }

    # Resultaten verzamelen     for ($i =
0; $i -lt $jobs.Count; $i++) {        try {
            $outputRaw = $powershells[$i].EndInvoke($jobs[$i])
            $powershells[$i].Dispose()

            $global:currentAfstand = $i + 1
            $global:progressBar.Value = [math]::Min($global:currentAfstand,
$global:progressBar.Maximum)
            $global:statusLabel.Text = "Analyzing... Afstand $($global:currentAfstand) van
$($global:progressBar.Maximum)"

            if ($outputRaw -and $outputRaw.Count -gt 0) {
                $output = $outputRaw[0]

                # Voortgang bijwerken
if ($output.Afstand) {
                    $global:currentAfstand = $output.Afstand
                }

                if ($output.Posities.Count -gt 0) {
                    $global:results.Add($output) | Out-Null
                }
            } else {
                [System.Windows.Forms.MessageBox]::Show(
                    "Thread $i → NO output or empty collection!",
                    "Error",
                    [System.Windows.Forms.MessageBoxButtons]::OK,
                    [System.Windows.Forms.MessageBoxIcon]::Error
                )
exit for            }
        } catch {
            [System.Windows.Forms.MessageBox]::Show(
                "ERROR thread $i → $_",
                "Error",
                [System.Windows.Forms.MessageBoxButtons]::OK,
                [System.Windows.Forms.MessageBoxIcon]::Error
            )
exit for
        }
    }

    $runspacePool.Close()
    $runspacePool.Dispose()
```

```
}

# ^^^^^ Copy before, for AI Check, Checkblock 2 ^^^^^
# Show Chart: Blok 7 function
Show-MatchChartPopup {
param (
      [Parameter(Mandatory = $true)]
      [System.Collections.ArrayList]$results
   )

   # Bereken totaal aantal matches
   $totalMatches = ($results | ForEach-Object { $_.Posities.Count } | Measure-Object -Sum).Sum
if (-not $totalMatches -or $totalMatches -eq 0) {
[System.Windows.Forms.MessageBox]::Show(
         "No matches availible for visualisation.",
         "Empty GraphChart",
         [System.Windows.Forms.MessageBoxButtons]::OK,
         [System.Windows.Forms.MessageBoxIcon]::Error
      )
return
   }

   $filteredResults = $results | Where-Object {
      $_ -ne $null -and $_.Posities -ne $null -and $_.Posities.Count -gt 0
   }

   $percentageSeries = New-Object System.Windows.Forms.DataVisualization.Charting.Series
   $percentageSeries.Name = "Matchpercentage"
   $percentageSeries.ChartType = "Column"
   $percentageSeries.Color = 'SteelBlue'
   $percentageSeries.LegendText = "Count per letterdistance"

   foreach ($item in $filteredResults) {
      $percentage = [math]::Round(100 * $item.Posities.Count / $totalMatches, 2)
      $x = "$($item.Afstand)"
      $pointIndex = $percentageSeries.Points.AddXY($x, $percentage)
       $percentageSeries.Points[$pointIndex].ToolTip = "$($item.Posities.Count) matches
($percentage%)"
       $percentageSeries.Points[$pointIndex].Label = "$percentage`% — $($item.Posities.Count)
match(es)"
      $percentageSeries.Points[$pointIndex].ToolTip = "$($item.Posities.Count) matches"
      $percentageSeries.Points[$pointIndex].Label = "$($item.Posities.Count) match(es)"
      $percentageSeries.Points[$pointIndex].LabelForeColor = 'DarkBlue'
      $percentageSeries.Points[$pointIndex].Font = 'Segoe UI, 8pt'
   }

   # Grafiekgebied en assen
   $chartArea = New-Object System.Windows.Forms.DataVisualization.Charting.ChartArea
   $chartArea.AxisX.Title = "Letter distance"
   $chartArea.AxisY.Title = "Percentage van totaal (%)"
```

```
$chartArea.AxisY.Minimum = 0
$chartArea.AxisY.Maximum = 100
$chartArea.AxisY.Interval = 10
$chartArea.AxisX.Interval = 1
$chartArea.AxisX.ScaleView.Zoomable = $true
$chartArea.CursorX.IsUserEnabled = $true
$chartArea.CursorX.IsUserSelectionEnabled = $true
$chartArea.AxisY.ScaleView.Zoomable = $true
$chartArea.CursorY.IsUserEnabled = $true
$chartArea.CursorY.IsUserSelectionEnabled = $true
$chartArea.AxisX.ScrollBar.Enabled = $true
$chartArea.AxisX.ScaleView.MinSize = 1  # voorkomt overzoom

# Grafiekcomponent
$chart = New-Object System.Windows.Forms.DataVisualization.Charting.Chart
$chart.Dock = 'Fill'  # Maakt meeschalend
$chart.Series.Add($percentageSeries)
$chart.ChartAreas.Add($chartArea)
$chart.Legends.Add("Legend")
$chart.Add_Click({
    $chart.ChartAreas[0].AxisX.ScaleView.ZoomReset()
    $chart.ChartAreas[0].AxisY.ScaleView.ZoomReset()
})

# Venster
$form = New-Object System.Windows.Forms.Form
$form.Text = "Matches per letter (percentage)"
$form.WindowState = "Normal"
$form.Size = '1920,1080'
$form.StartPosition = "CenterScreen"
$form.FormBorderStyle = "Sizable"
$form.MinimumSize = '600,400'
$form.Controls.Add($chart)
$form.TopMost = $true

$resetZoomBtn = New-Object System.Windows.Forms.Button
$resetZoomBtn.Text = "Reset Zoom"
$resetZoomBtn.Dock = 'Bottom'
$resetZoomBtn.Height = 30
$resetZoomBtn.Add_Click({
    $chart.ChartAreas[0].AxisX.ScaleView.ZoomReset()
    $chart.ChartAreas[0].AxisY.ScaleView.ZoomReset()
})
$form.Controls.Add($resetZoomBtn)

$exportBtn = New-Object System.Windows.Forms.Button
$exportBtn.Text = "Export RaphChart *.png; *.jpg; *.jpeg; *.bmp; *.pdf"
$exportBtn.Dock = 'Bottom'
$exportBtn.Height = 30
$exportBtn.Add_Click({
```

```
        $SaveFileDialog = New-Object System.Windows.Forms.SaveFileDialog
        $SaveFileDialog.InitialDirectory = $PSScriptRoot
        $SaveFileDialog.Filter = "ImageFiles (*.png; *.jpg; *.jpeg; *.bmp; *.pdf)|*.png; *.jpg; *.jpeg;
*.bmp; *pdf|All Files (*.*)|*.*"
        $SaveFileDialog.Title = "Save as..."
        $SaveFileDialog.DefaultExt = "jpg"
        $SaveFileDialog.AddExtension = $true
        $SaveFileDialog.OverwritePrompt = $true

        if ($SaveFileDialog.ShowDialog() -eq "OK") {
            $pad = $SaveFileDialog.FileName
            $extensie = [System.IO.Path]::GetExtension($pad).ToLower()
            $naamZonderExt = [System.IO.Path]::GetFileNameWithoutExtension($pad)
            $map = Split-Path $pad -Parent
◌
            switch ($extensie) {
                ".png" { $formaat =
[System.Windows.Forms.DataVisualization.Charting.ChartImageFormat]::Png }
                ".jpg" { $formaat =
[System.Windows.Forms.DataVisualization.Charting.ChartImageFormat]::Jpeg }
                ".jpeg" { $formaat =
[System.Windows.Forms.DataVisualization.Charting.ChartImageFormat]::Jpeg }
                ".bmp" { $formaat =
[System.Windows.Forms.DataVisualization.Charting.ChartImageFormat]::Bmp }
                ".pdf" { $formaat =
[System.Windows.Forms.DataVisualization.Charting.ChartImageFormat]::Pdf }
default {
                    [System.Windows.Forms.MessageBox]::Show(
                        "Unknown format: $extensie",
                        "Error",
                        [System.Windows.Forms.MessageBoxButtons]::OK,
                        [System.Windows.Forms.MessageBoxIcon]::Error
                    )
return
                }
            }

            if (Test-Path $pad) {
                $keuze = [System.Windows.Forms.MessageBox]::Show(
                    "File exsists:`n$pad`n`nWil je het overschrijven?",
                    "File exsists",
                    [System.Windows.Forms.MessageBoxButtons]::YesNoCancel,
                    [System.Windows.Forms.MessageBoxIcon]::Question
                )

                switch ($keuze) {
                    'Yes' {
                        # Gewoon overschrijven
                        $chart.DataBind()
                        $chart.SaveImage($pad, $formaat)
```

```
            }
            'No' {
                # Maak een kopie-bestandsnaam
                $kopiePad = Join-Path $map ("${naamZonderExt}_kopie$extensie")
                $i = 2
                while (Test-Path $kopiePad) {
                    $kopiePad = Join-Path $map ("${naamZonderExt}_kopie$i$extensie")
                    $i++
                }

                $chart.SaveImage($kopiePad, $formaat)

                [System.Windows.Forms.MessageBox]::Show(
                    "GraphChart saved as:`n$kopiePad",
                    "Copie made",
                    [System.Windows.Forms.MessageBoxButtons]::OK,
                    [System.Windows.Forms.MessageBoxIcon]::Information
                )
            }
            'Cancel' {
                # Gebruiker annuleert
                [System.Windows.Forms.MessageBox]::Show(
                    "Save Cancelled.",
                    "Cancelled",
                    [System.Windows.Forms.MessageBoxButtons]::OK,
                    [System.Windows.Forms.MessageBoxIcon]::Information
                )
return
            }
        }
    } else {
        # Bestand bestaat nog niet — gewoon opslaan
        $chart.DataBind()
        $chart.SaveImage($pad, $formaat)
    }

    [System.Windows.Forms.MessageBox]::Show(
        "GraphChart saved as:`n$pad",
        "$formaat Export",
        [System.Windows.Forms.MessageBoxButtons]::OK,
        [System.Windows.Forms.MessageBoxIcon]::Information
    )
} else {
    [System.Windows.Forms.MessageBox]::Show(
        "Dave cancelled.",
        "Error",
        [System.Windows.Forms.MessageBoxButtons]::OK,
        [System.Windows.Forms.MessageBoxIcon]::Information
    )
}
```

```
    })
    $form.Controls.Add($exportBtn)

    # Eventueel: meeschalende fonts per resize
    $form.Add_Resize({
        $chart.Font = New-Object System.Drawing.Font("Segoe UI", [int]($form.Height / 48))
    })

    $form.ShowDialog()
}

# ^^^^^ Copy before, for AI Check, Checkblock 3 ^^^^^

# Toon LogViewer: Blok 8
function Toon-LogViewer {
param (
        [string]$pad
    )

    if (-not (Test-Path $pad)) {
        [System.Windows.Forms.MessageBox]::Show(
            "Logfile not found:`n$pad",
            "Error",
            [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error
        )
return
    }

    $formLog = New-Object System.Windows.Forms.Form
    $formLog.Text = "View Analysis-log"
    $formLog.Size = '600,400'
    $formLog.StartPosition = 'CenterScreen'

    $logBox = New-Object System.Windows.Forms.TextBox
    $logBox.Multiline = $true
    $logBox.ReadOnly = $true
    $logBox.ScrollBars = "Vertical"
    $logBox.Dock = 'Fill'
    $logBox.Font = 'Consolas, 10pt'
    $formLog.Controls.Add($logBox)

    $inhoud = Get-Content $pad -Raw
    $logBox.Text = $inhoud

    $formLog.ShowDialog()
}

#  Lees-ZuivereInhoud: Blok 9 function
Lees-ZuivereInhoud {
```

```
    param([string]$pad)

    # Lees de ruwe inhoud
    $inhoud = Get-Content $pad -Raw

    # Normaliseer eerst
    $genormaliseerd = $inhoud.Normalize([Text.NormalizationForm]::FormD)

    # Verwijder diakritische tekens (Mn = Nonspacing Mark)
    $zonderDiakritiek = [regex]::Replace($genormaliseerd, '\p{Mn}', '')

    # Nu pas: verwijder whitespace, newline, maqaf enz.
if ($spaceCheckBox.Checked) {
        $schoon = $zonderDiakritiek -replace "[\s`r`n־]", ""
    } else {
         $schoon = $zonderDiakritiek
    }

    return $schoon
}

# Genereer-HTMLVisualisatie: Blok 10 function
Genereer-HTMLVisualisatie {
    param (
        [string]$inhoudPad,
        [hashtable]$positieAfstanden,
        [string]$outputPad,
        [int]$regelBreedte = 80,
        [int]$woordLengte = 5,
        [bool]$GebruikRegelBreedte = $false
    )

    $tekst = Get-Content $inhoudPad -Raw
    $letters = $tekst.ToCharArray()
    $htmlBody = ""
    $uniekeAfstanden = @{}
    $afstandIndex = @{}

    for ($i = 0; $i -lt $letters.Length; $i++) {
        $char = $letters[$i]        if
($positieAfstanden.ContainsKey($i)) {
$afstanden = $positieAfstanden[$i]
            if ($afstanden -isnot [System.Collections.IEnumerable]) {
                $afstanden = @($afstanden)
            }

            foreach ($a in $afstanden) {
$uniekeAfstanden[$a] = $true
                if (-not $afstandIndex.ContainsKey($a)) { $afstandIndex[$a] = 0 }
            }
```

```
            $afstandStr = $afstanden -join ' '
            $title = "Afstanden: " + ($afstanden -join ', ')
            $ids = @()
            foreach ($a in $afstanden) {
                $ids += "match-$a-$($afstandIndex[$a])"
                $afstandIndex[$a]++
            }
            $idStr = $ids[0]

            $htmlBody += "<span class='match' data-afstand='$afstandStr' title='$title'
id='$idStr'>$char</span>"
        } else {
            $htmlBody += $char
        }

        if ($GebruikRegelBreedte -and (($i + 1) % $regelBreedte -eq 0)){
            $htmlBody += "`n"
        }
    }

    $dropdown = "<select id='afstandFilter' onchange='filterAfstand()'>`n"     $dropdown += "<option
    value="">-- No filter / Show all --</option>`n"     foreach ($afstand in ($uniekeAfstanden.Keys |
    Sort-Object)) {
        $dropdown += "<option value='$afstand'>Distance $afstand</option>`n"
    }
    $dropdown += "</select>"
    $htmlBody = "<pre>`n$htmlBody`n</pre>"

$html = @"
<!DOCTYPE html>
<html lang='he'>
<head>
    <meta charset='UTF-8'>
    <title>Matrix Visualisatie</title>
    <style>         #bediening {
position: sticky;             top: 0;
background-color: white;
padding: 10px;             z-index:
10;
        }         body {             font-family:
'Courier New', monospace;
            font-size: 16px;
direction: rtl;
        }
        pre {
            white-space: pre-wrap;
line-height: 1.4;
        }
        .match {
```

```
            color: red;
            background-color: yellow;
        }
        .match.highlighted {            color:
white !important;            background-
color: red !important;
            font-weight: bold !important;
        }
    </style>
<script>
        const woordLengte = $woordLengte;
    </script>

</head>
<body>
<h2>Spronganalyse Visualisatie</h2>

<div id="bediening">
    $dropdown
    <button onclick="volgendeMatch()">Next</button>
    <button onclick="vorigeMatch()">Previous</button>
    <span id="matchTeller">Match 1 of 0</span>
</div>
$htmlBody
<script> let
huidigeIndex = 0;
let huidigeAfstand = "";

function filterAfstand() {     huidigeAfstand =
document.getElementById('afstandFilter').value.trim();     huidigeIndex =
0;

    const matches = document.querySelectorAll('.match');
matches.forEach(el => {         const afstanden =
el.dataset.afstand.split(' ').map(a => a.trim());         if
(huidigeAfstand && afstanden.includes(huidigeAfstand)) {
el.classList.add('highlighted');
        } else {
            el.classList.remove('highlighted');
        }
    });     updateTeller();
scrollNaarMatch(huidigeAfstand, huidigeIndex);
}

function scrollNaarMatch(afstand, index) {     const el =
document.getElementById("match-" + afstand + "-" + index);     if (el) {
el.scrollIntoView({ behavior: 'smooth', block: 'center' });
    }
}
```

```
function volgendeMatch() {     huidigeIndex += $woordLengte;     const el =
document.getElementById("match-" + huidigeAfstand + "-" + huidigeIndex);     if (el) {
el.scrollIntoView({ behavior: 'smooth', block: 'center' });
   } else {         huidigeIndex = 0;
scrollNaarMatch(huidigeAfstand, huidigeIndex);
   }
   updateTeller();
}

function vorigeMatch() {     huidigeIndex -= $woordLengte;     const el =
document.getElementById("match-" + huidigeAfstand + "-" + huidigeIndex);     if (el) {
el.scrollIntoView({ behavior: 'smooth', block: 'center' });
   } else {         huidigeIndex = 0;
scrollNaarMatch(huidigeAfstand, huidigeIndex);
   }
   updateTeller();
}

function updateTeller() {
   const teller = document.getElementById("matchTeller"); if
   (!teller) return;

   const total = document.querySelectorAll(".match.highlighted").length;
   if (total === 0) {
teller.textContent = "Geen matches";
   } else {
      teller.textContent = "Match " + Math.ceil((huidigeIndex + 1) / woordLengte) + " of " + total /
$woordLengte;
   }
}
</script>
</body>
</html>
"@


   Set-Content -Path $outputPad -Value $html -Encoding UTF8
   Start-Process $outputPad
}

# Genereer-HTMLVisualisatie: Blok 11
function Bouw-PositieAfstanden {     param
([array]$results, [string]$zoekterm)

   $positieAfstanden = @{}     foreach
($result in $results) {         $afstand =
$result.Afstand         foreach ($start in
$result.Posities) {
         for ($i = 0; $i -lt $zoekterm.Length; $i++) {
            $index = $start + ($i * $afstand)
            if (-not $positieAfstanden.ContainsKey($index)) {
```

```
                $positieAfstanden[$index] = @()
            }
            if (-not ($positieAfstanden[$index] -contains $afstand)) {
                $positieAfstanden[$index] += $afstand
            }
        }
    }
    }
    return $positieAfstanden
}

# ^^^^^ Copy before, for AI Check ,Checkblock 4 ^^^^^

# GUI en CSV-analyse: Blok 12 function
Launch-GuiWithCsv {
    $toolTip = New-Object System.Windows.Forms.ToolTip
    $form = New-Object System.Windows.Forms.Form
    $form.Text = "CSV Analysis"
    $form.Size = '600,400'
    $form.StartPosition = 'CenterScreen'

    $label = New-Object System.Windows.Forms.Label
    $label.Text = "Enter search word:"
    $label.Location = '16,14'
    $label.AutoSize = '300,20'
    $label.TextAlign = 'MiddleCenter'
    $form.Controls.Add($label)

    $textBox = New-Object System.Windows.Forms.TextBox
    $textBox.Text = "אלהים"
    $textBox.Size = '300,27'
    $textBox.Location = '20,42'
    $textBox.TextAlign = 'Center'
    $form.Controls.Add($textBox)

    # Exportknop
    $exportButton = New-Object System.Windows.Forms.Button
    $exportButton.Text = "Export"
    $exportButton.Size = '93,29'
    $exportButton.Location = '322,10'
    $exportButton.Enabled = $false
    $exportButton.TextAlign = 'MiddleCenter'
    $form.Controls.Add($exportButton)

    # Label voor threads
    $threadsLabel = New-Object System.Windows.Forms.Label
    $threadsLabel.Text = "Threads:"
    $threadsLabel.Location = '506,14'   # rechts onder "Zoek in CSV"
    $threadsLabel.Size = '64,20'
    $threadsLabel.TextAlign = 'MiddleCenter'
```

```
$form.Controls.Add($threadsLabel)

$toolTip.SetToolTip($threadsLabel, "Nr. of Threads (CPU cores) for analysis.")

# Zoek in CSV knop
$button = New-Object System.Windows.Forms.Button
$button.Text = "Search CSV"
$button.Size = '93,29'
$button.Location = '322,39'
$button.TextAlign = 'MiddleCenter'
$form.Controls.Add($button)

# Label voor maximale sprongafstand
$maxAfstandLabel = New-Object System.Windows.Forms.Label
$maxAfstandLabel.Text = "Max distance letters"
$maxAfstandLabel.Size = '160,20'
$maxAfstandLabel.Location = '20,72'
$maxAfstandLabel.TextAlign = 'MiddleRight'
$form.Controls.Add($maxAfstandLabel)

$toolTip.SetToolTip($maxAfstandLabel, "Enter max for maximum.")

# Invoerveld voor sprongafstandwaarde
$maxAfstandBox = New-Object System.Windows.Forms.TextBox
$maxAfstandBox.Text = "1000"
$maxAfstandBox.Size = '134,27'
$maxAfstandBox.Location = '186,70'
$maxAfstandBox.TextAlign = 'Center'
$form.Controls.Add($maxAfstandBox)

$toolTip.SetToolTip($maxAfstandBox, "Enter max for maximum.")

# Label voor maximale sprongafstand
$filterLengteLabel = New-Object System.Windows.Forms.Label
$filterLengteLabel.Text = "HTML block length:"
$filterLengteLabel.Size = '150,20'
$filterLengteLabel.Location = '166,100'
$filterLengteLabel.TextAlign = 'MiddleRight'
$form.Controls.Add($filterLengteLabel)

# Invoerveld voor sprongafstandwaarde
$filterLengteBox = New-Object System.Windows.Forms.TextBox
$filterLengteBox.Text = "80"
$filterLengteBox.Size = '93,27'
$filterLengteBox.Location = '322,97'
$filterLengteBox.TextAlign = 'Center'
$filterLengteBox.Enabled = $false
$form.Controls.Add($filterLengteBox)

# Label voor maximale sprongafstand
```

```
$analysisLabel = New-Object System.Windows.Forms.Label
$analysisLabel.Text = "Analysis"
$analysisLabel.Size = '102,20'
$analysisLabel.Location = '421,77'
$analysisLabel.TextAlign = 'MiddleRight'
$form.Controls.Add($analysisLabel)

# Invoerveld voor sprongafstandwaarde
$analysisCheckBox = New-Object System.Windows.Forms.CheckBox
$analysisCheckBox.Name = "Analysis"
$analysisCheckBox.Size = '18,17'
$analysisCheckBox.Location = '529,79'
$form.Controls.Add($analysisCheckBox)

# Label voor maximale sprongafstand
$grafiekLabel = New-Object System.Windows.Forms.Label
$grafiekLabel.Text = "Graph"
$grafiekLabel.Size = '102,20'
$grafiekLabel.Location = '421,100'
$grafiekLabel.TextAlign = 'MiddleRight'
$form.Controls.Add($grafiekLabel)

# Invoerveld voor sprongafstandwaarde
$grafiekCheckBox = New-Object System.Windows.Forms.CheckBox
$grafiekCheckBox.Name = "Grafiek"
$grafiekCheckBox.Size = '18,17'
$grafiekCheckBox.Location = '529,102'
$form.Controls.Add($grafiekCheckBox)

# Label voor maximale sprongafstand
$htmlLabel = New-Object System.Windows.Forms.Label
$htmlLabel.Text = "HTML Filter"
$htmlLabel.Size = '88,20'
$htmlLabel.Location = '421,14'
$htmlLabel.TextAlign = 'MiddleCenter'
$form.Controls.Add($htmlLabel)

# Invoerveld voor sprongafstandwaarde
$htmlCheckBox = New-Object System.Windows.Forms.CheckBox
$htmlCheckBox.Name = "HTML"
$htmlCheckBox.Size = '18,17'
$htmlCheckBox.Location = '456,45'
$form.Controls.Add($htmlCheckBox)

# Log bekijken knop
$logButton = New-Object System.Windows.Forms.Button
$logButton.Text = "Log"
$logButton.Size = '93,29'
$logButton.Location = '322,68'  # onder 'Zoek in CSV'
$logButton.TextAlign = 'MiddleCenter'
```

```
    $form.Controls.Add($logButton)

    # NumericUpDown voor aantal threads
    $threadsBox = New-Object System.Windows.Forms.NumericUpDown
    $threadsBox.Minimum = 1
    $threadsBox.Maximum = ([Environment]::ProcessorCount)
    $threadsBox.Value = 1               # standaard op single-thread
    $threadsBox.Location = '506,43'
    $threadsBox.Size = '64,27'
    $threadsBox.TextAlign = 'Center'
    $form.Controls.Add($threadsBox)

    $toolTip.SetToolTip($threadsBox, "Nr. of Threads (CPU cores) for analysis.")

    $newlineLabel = New-Object System.Windows.Forms.Label
    $newlineLabel.Text = "Remove Spaces:"
    $newlineLabel.Size = '120,20'
    $newlineLabel.Location = '20,100'
    $newlineLabel.TextAlign = 'MiddleCenter'
    $form.Controls.Add($newlineLabel)
    $toolTip.SetToolTip($newlineLabel, "Remove Maqaf and all spaces between all letters, the whole
scripture as one continuouse word.")

    $spaceCheckBox = New-Object System.Windows.Forms.CheckBox
    $spaceCheckBox.Name = "Space"
    $spaceCheckBox.Size = '18,17'
    $spaceCheckBox.Location = '150,102'
    $form.Controls.Add($spaceCheckBox)
    $toolTip.SetToolTip($spaceCheckBox, "Remove Maqaf and all spaces between all letters, the whole
scripture as one continuouse word.")
    $spaceCheckBox.Add_CheckedChanged({
       $filterLengteBox.Enabled = $spaceCheckBox.Checked
    })
    # Groep voor resultaten
    $groupBox = New-Object System.Windows.Forms.GroupBox
    $groupBox.Text = "Matchresults"
    $groupBox.Size = '539,151'
    $groupBox.Location = '23,133'
    $form.Controls.Add($groupBox)

    # Multiline TextBox binnen GroupBox
    $resultBox = New-Object System.Windows.Forms.TextBox
    $resultBox.Multiline = $true
    $resultBox.ReadOnly = $true
    $resultBox.ScrollBars = "Vertical"
    $resultBox.Size = '527,128'
    $resultBox.Location = '6,17'
    $groupBox.Controls.Add($resultBox)

    # $global:statusLabel = New-Object System.Windows.Forms.Label
```

```
    $global:statusLabel.Location = '16,297'
    $global:statusLabel.Size = '554,20'
    $global:statusLabel.Text = ""
    $global:statusLabel.TextAlign = 'MiddleCenter'
    $form.Controls.Add($global:statusLabel)

    # Voortgangsbalk
    $maxAfstand = 1000
    $global:progressBar.Location = '16,320'
    $global:progressBar.Size = '554,15'
    $global:progressBar.Style = 'Blocks'
    $global:progressBar.Minimum = 1
    $global:progressBar.Maximum = $maxAfstand
    $global:progressBar.Value = 1
    $form.Controls.Add($global:progressBar)

    $button.Add_Click({
                      Clear-mem  #   Alles leegmaken en GUI opfrissen

                    # Timer voor GUI-updates tijdens threads
                    $timer = New-Object System.Windows.Forms.Timer
                    $timer.Interval = 200
$timer.Add_Tick({
                            if ($global:currentAfstand) {
                                    $global:progressBar.Value = [math]::Min($global:currentAfstand,
$global:progressBar.Maximum)
                                    $global:statusLabel.Text = "Analyzing... Afstand
$($global:currentAfstand) van $($global:progressBar.Maximum)"
                                    [System.Windows.Forms.Application]::DoEvents()
                            }
                    })
                    $timer.Start()

        $global:csvFolder = ""
        $global:result = ""
        $global:zoekterm = $textBox.Text
        $exportButton.Enabled = $false
        $form.Refresh()

        if ([string]::IsNullOrWhiteSpace($global:zoekterm)) {
           [System.Windows.Forms.MessageBox]::Show(
              "Search em.pty or invalid",
              "Error",
              [System.Windows.Forms.MessageBoxButtons]::OK,
              [System.Windows.Forms.MessageBoxIcon]::Error
           )
return
        }

        $letters = $global:zoekterm.ToCharArray()
```

```
        $OpenFileDialog = New-Object System.Windows.Forms.OpenFileDialog
        $OpenFileDialog.InitialDirectory = $PSScriptRoot
        $OpenFileDialog.Filter = "CSV-Files (*.csv)|*.csv|All Files (*.*)|*.*"
        $OpenFileDialog.Title = "Selecteer een CSV-bestand"

        if ($OpenFileDialog.ShowDialog() -eq "OK") {
           $global:csvPad = $OpenFileDialog.FileName
        } else {
           [System.Windows.Forms.MessageBox]::Show(
              "No File selected.",
              "Error",
              [System.Windows.Forms.MessageBoxButtons]::OK,
              [System.Windows.Forms.MessageBoxIcon]::Information
           )
                                return
        }
        $global:csvFolder = Split-Path $global:csvPad

        #   Verwijder aanhalingstekens en extra spaties
        $csvPadClean = $global:csvPad.Trim().Trim("")

        if (-not (Test-Path $csvPadClean)) {
           [System.Windows.Forms.MessageBox]::Show(
              "CSV-File not found!",
              "Error",
              [System.Windows.Forms.MessageBoxButtons]::OK,
              [System.Windows.Forms.MessageBoxIcon]::Information
           )
return
        }

# ^^^^^ Copy before, for AI Check ,Checkblock 5 ^^^^^

        # Gebruik het opgeschoonde pad verderop
        $global:csvPad = $csvPadClean
        $inhoud = Lees-ZuivereInhoud $csvPad
        $bestandsNaam = Split-Path $global:csvPad -Leaf

        if ($spaceCheckBox.Checked) {
           if ($bestandsNaam -notlike "*Clean_*") {
              $cleanPath = Join-Path $global:csvFolder ("Clean_" + $bestandsNaam)
if (-not (Test-Path $cleanPath)) {
                 $inhoud | Out-File -FilePath $cleanPath -Encoding UTF8BOM
              }
           }
        }

        $inhoud = $inhoud.ToCharArray()
```

```
        if ($inhoud.Length -eq 0) {
            [System.Windows.Forms.MessageBox]::Show(
                "CSV filterable letters.",
                "No letters",
                [System.Windows.Forms.MessageBoxButtons]::OK,
                [System.Windows.Forms.MessageBoxIcon]::Information
            )
return
        }

        $letterPosities = @{}

        foreach ($letter in $letters) {
$posities = @()
            for ($i = 0; $i -lt $inhoud.Length; $i++) {
                if ($inhoud[$i] -eq $letter) {
                    $posities += $i
                }
            }
            $letterPosities[$letter] = $posities
        }

        # Toon individuele letterposities
$global:results = @()
        foreach ($kvp in $letterPosities.GetEnumerator()) {
            $lijst = ($kvp.Value -join ", ")
            $global:result += "Letter '$($kvp.Key)' at positions: $lijst`n"
        }

        $analyseSW = [System.Diagnostics.Stopwatch]::StartNew()

        $global:results = @()

        # Multithreaded spronganalyse
$maxThreads = $threadsBox.Value         if
($maxThreads -le 1) {
            $inhoudLength = $inhoud.Length
            $woordLength = $global:zoekterm.Length
            $maxAfstand = $maxAfstandBox.Text
            If ($maxAfstand -eq "max") {
                $maxAfstand = [math]::Floor($inhoudLength / ($woordLength - 1))
            } else {
                $maxAfstand = [int]$maxAfstandBox.Text
            }
            $global:progressBar.Maximum = $maxAfstand

            for ($afstand = 1; $afstand -le $maxAfstand; $afstand++) {

                $global:progressBar.Value = $afstand
```

```
            $global:statusLabel.Text = "Analyzing... Afstand $afstand van $maxAfstand"
[System.Windows.Forms.Application]::DoEvents()  # zorgt dat GUI bijwerkt!

            $maxStart = $inhoudLength - (($woordLength - 1) * $afstand) - 1
            $matches = New-Object 'System.Collections.Generic.List[int]'

            for ($start = 0; $start -le $maxStart; $start++) {
               $gevonden = $true
               for ($i = 0; $i -lt $woordLength; $i++) {
$pos = $start + ($i * $afstand)
                  if ($inhoud[$pos] -ne $global:zoekterm[$i]) {
                     $gevonden = $false
                     break
                  }
               }
               if ($gevonden) {
                  $matches.Add($start)
               }
            }

            if ($matches.Count -gt 0) {
               $global:results += [PSCustomObject]@{
                  Afstand  = $afstand
                  Posities = $matches
               }
            }
         }
                                 $global:positieAfstanden = @{}
  $global:positieAfstanden = Bouw-PositieAfstanden -results $global:results zoekterm
$global:zoekterm
      } else {
         $inhoudLength = $inhoud.Length
         $woordLength = $global:zoekterm.Length
         $maxAfstand = $maxAfstandBox.Text
         If ($maxAfstand -eq "max") {
            $maxAfstand = [math]::Floor($inhoudLength / ($woordLength - 1))
         } else {
            $maxAfstand = [int]$maxAfstandBox.Text
         }
         $global:progressBar.Maximum = $maxAfstand

         Analyse-SprongMultithread -zoekterm $global:zoekterm.ToCharArray() -inhoud $inhoud
inhoudLength $inhoud.Length -woordLength $global:zoekterm.Length -maxAfstand $maxAfstand
maxThreads $maxThreads
         $global:positieAfstanden = @{}
         $global:positieAfstanden = Bouw-PositieAfstanden -results $global:results -zoekterm
$global:zoekterm
      }

                  if ($global:timer -and $global:timer.Enabled) {
```

```
                    $global:timer.Stop()
                    Remove-Variable -Name timer -Scope Global -ErrorAction SilentlyContinue
                }

    if ($htmlCheckBox.Checked){
        Genereer-HTMLVisualisatie -inhoudPad $global:csvPad `
                    -positieAfstanden $global:positieAfstanden `
                    -outputPad (Join-Path (Split-Path $global:csvPad -Parent)
("${bestandsNaam}_$($global:zoekterm)_Filter.html")) `
                    -regelBreedte ([int]$filterLengteBox.Text) `
                    -woordLengte $woordLength `
                    -GebruikRegelBreedte $spaceCheckBox.Checked
    }

    if ($analysisCheckBox.Checked){
        $duration = "{0:N2}" -f $analyseSW.Elapsed.TotalSeconds
        $maxAfstandMatch = ($global:results | Select-Object -ExpandProperty Afstand |
MeasureObject -Maximum).Maximum
        $totalMatches = ($global:results | ForEach-Object { $_.Posities.Count } | Measure-Object
Sum).Sum

        $log = @()
        $log += "Analysis completed on: $(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')"
        $log += "Number of Treads used: $($threadsBox.Value)"
        $log += "Duration: $duration seconds"
        $log += "Longest letter spacing: $maxAfstandMatch"
        $log += "Total matches: $totalMatches"
        $log += "--------------------------------------"

        $timestamp = (Get-Date -Format "ddMMyyyy")
        $logFileName = "$global:zoekterm analysis $timestamp.txt"
        $logPath = Join-Path $global:csvFolder $logFileName
        Add-Content -Path $logPath -Value $log
    }

    if($grafiekCheckBox.Checked){
#   Forceer repaint vóór je popup
        Update-ResultBoxSafely -box $resultBox -results $global:results

        if ($global:results -and $global:results.Count -gt 0) {
            Show-MatchChartPopup -results  $global:results

            # Herstel GUI visueel erna
            Update-ResultBoxSafely -box $resultBox -results $global:results

        } else {
            [System.Windows.Forms.MessageBox]::Show(
                "No matches found to visualize.",
                "No GraphChart",
                [System.Windows.Forms.MessageBoxButtons]::OK,
```

```
            [System.Windows.Forms.MessageBoxIcon]::Information
        )
    }
}

# Clear-mem  #   Alles leegmaken en GUI opfrissen
$analyseSW.Stop()

# Directe woordmatch (afstand 1, letter na letter)
$matchesDirect = @()        for ($i = 0; $i -le
$inhoudLength - $woordLength; $i++) {           $substring =
$inhoud[$i..($i + $woordLength - 1)] -join ''           if
($substring -eq $global:zoekterm) {
            $matchesDirect += $i
        }
}

# Resultaat tonen inclusief directe matches
$report = "Search word: $global:zoekterm`n"

$teller = 1
if ($matchesDirect.Count -gt 0) {
foreach ($directPos in $matchesDirect) {
            $report += "$teller. Positie: $directPos (afstand: 1 — compleet word found)`n"
$teller++
        }
}

if ($global:results.Count -gt 0) {            foreach ($item in
($global:results | Sort-Object Afstand)) {                foreach ($pos
in $item.Posities) {
            $report += "$teller. Position: $pos (afstand: $($item.Afstand))`n"
$teller++
            }
        }
}

$exportButton.Enabled = $true
})

# ^^^^^ Copy before, for AI Check ,Checkblock 6 ^^^^^

$exportButton.Add_Click({
    $savePath = Join-Path $global:csvFolder "$global:zoekterm matches.txt"
if (-not [string]::IsNullOrWhiteSpace($savePath)) {           try {
            $resultBox.Text | Out-File -FilePath $savePath -Encoding UTF8BOM
            [System.Windows.Forms.MessageBox]::Show(
                "Results succesfull saved as UTF8BOM:`n$savePath",
                "Export ready",
                [System.Windows.Forms.MessageBoxButtons]::OK,
```

```
            [System.Windows.Forms.MessageBoxIcon]::Information
        )
    } catch {
        [System.Windows.Forms.MessageBox]::Show(
            "File not saved:`n$_",
            "Error",
            [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error
        )
    }
  }
})

# Klikactie om log te tonen
$logButton.Add_Click({

  # Alternatief: pak laatste gemaakte log als pad
  $recentLog = Get-ChildItem -Path $global:csvFolder -Filter "*analysis*.txt" |
               Sort-Object LastWriteTime -Descending |
               Select-Object -First 1

  if ($recentLog) {
    Toon-LogViewer -pad $recentLog.FullName
  } else {
    [System.Windows.Forms.MessageBox]::Show(
      "No log File found.",
      "Info",
      [System.Windows.Forms.MessageBoxButtons]::OK,
      [System.Windows.Forms.MessageBoxIcon]::Information
    )
  }
})

$laatsteMatch = ($global:results | ForEach-Object { $_.Posities } | Where-Object { $_ } |
SelectObject -Last 1)

if ($laatsteMatch) {
  $global:statusLabel.Text = "Last matchposition: $laatsteMatch"
} else {
  $global:statusLabel.Text = "No matches found"
}

$form.ShowDialog()
}

# Start: Blok 13
$keuze = Show-Menu switch
($keuze) {    "1" { Convert-
ExcelToCsv }    "2" {
```

```
Launch-GuiWithCsv }
default {
        [System.Windows.Forms.MessageBox]::Show(
            "Invalid choice.",
            "Invalid",
            [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error
        )
    }
}
```

 # ^^^^^ Copy before, for AI Check ,Checkblock 7 ^^^^^

Excel naar csv.ps1: _____

Letterlijk.bat: _____

```
@echo off
setlocal EnableDelayedExpansion

REM -- Detect Windows version -- for /f
"tokens=4-5 delims=. " %%i in ('ver') do (
   set "winVer=%%i.%%j"
)

REM -- Detect Sandbox -- set "isSandbox=false" echo
%USERPROFILE% | findstr /i "WDAGUtilityAccount" >nul if
!errorlevel! == 0 (
   set "isSandbox=true"
)

REM -- Define paths --
set "PS7Path=C:\Program Files\PowerShell\7" set
"PS7Exe=%PS7Path%\pwsh.exe" set
"DesktopPath=%USERPROFILE%\Desktop\PowerShell" set
"DesktopCSVPath=%USERPROFILE%\Desktop\CSV\Alfabet"
set "ZipPath=%DesktopPath%\PowerShell7"
set "zipFile=%DesktopPath%\PowerShell.zip"
set "msiFile=%TEMP%\pwsh7.msi" set
"pwshCmd="

if not exist "!DesktopPath!" (
mkdir "!DesktopPath!"
   mkdir "!DesktopCSVPath!"
)

REM -- Check if pwsh is available via PATH --
where pwsh >nul 2>nul if !errorlevel! == 0 (     echo
? PowerShell 7 is already available via PATH.
   set "pwshCmd=pwsh"
) else (
   if "!isSandbox!"=="true" (
      REM -- Check if pwsh.exe already exists in ZIP path --
if exist "!ZipPath!\pwsh.exe" (             echo ? PowerShell 7
already extracted in Sandbox.
         set "pwshCmd=!ZipPath!\pwsh.exe"
      ) else (
         echo ? Downloading PowerShell 7 ZIP for Sandbox...
powershell -Command "Invoke-WebRequest -Uri
'https://github.com/PowerShell/PowerShell/releases/download/v7.4.0/PowerShell-7.4.0-win-x64.zip'
OutFile '!zipFile!'"            powershell -Command "Expand-Archive -Path '!zipFile!' -DestinationPath
'!ZipPath!' -Force"             set "pwshCmd=!ZipPath!\pwsh.exe"
      )
   ) else (
      REM -- Host: Try WinGet first --         where
winget >nul 2>nul         if !errorlevel! == 0 (
```

```
echo ? Installing PowerShell 7 via WinGet...
winget install --id Microsoft.PowerShell --source
winget --accept-package-agreements -accept-
source-agreements
      ) else (
         echo ? WinGet not available. Downloading MSI installer...
         powershell -Command "Invoke-WebRequest -Uri
'https://github.com/PowerShell/PowerShell/releases/download/v7.4.0/PowerShell-7.4.0-win-x64.msi'
OutFile '!msiFile!'"
         msiexec /i "!msiFile!" /quiet /norestart
      )

      REM -- Verify installation --        if exist
"!PS7Exe!" (            echo ? PowerShell 7
installed successfully.
         set "pwshCmd=!PS7Exe!"

         REM -- Add to user PATH for Windows 10 and 11 --
echo ? Adding PowerShell 7 to user PATH...
         setx PATH "%PATH%;!PS7Path!" >nul
      ) else (            echo ? Installation failed.
Exiting script.
         goto :eof
      )
   )
)

REM -- Launch your PowerShell script -- echo
?? Launching EXCEL naar csv.ps1...
"!pwshExe!" -ExecutionPolicy Bypass -File "%USERPROFILE%\Desktop\Letterlijk.ps1"

Endlocal
```

Letterlijk.bat: _____

Letterlijk.ps1: _____

```
# ? Forceer adminrechten (werkt voor zowel PowerShell Core als Windows PowerShell) if
(-not ([Security.Principal.WindowsPrincipal] `
   [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole(`
   [Security.Principal.WindowsBuiltInRole]::Administrator)) {

   $exe = if ($PSVersionTable.PSEdition -eq 'Core') { 'pwsh.exe' } else { 'powershell.exe' }

   Start-Process $exe -Verb RunAs -ArgumentList @(
      "-NoProfile",
      "-ExecutionPolicy", "Bypass",
      "-File", "`"$PSCommandPath`""
```

```
    )
exit
}

# Explicit execution policy for current process
Set-ExecutionPolicy Bypass -Scope Process
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# Dialoogvenster om CSV-bestand te kiezen
$openFileDialog = New-Object System.Windows.Forms.OpenFileDialog
$openFileDialog.Filter = "CSV Files (*.csv)|*.csv"
$openFileDialog.Title = "Select a CSV-File"
$openFileDialog.InitialDirectory =  Join-Path $PSScriptRoot "CSV"
$null = $openFileDialog.ShowDialog()

if (-not $openFileDialog.FileName) {     Write-
Host "No file selected."
    return
}

$fntSize = 60
$picSize = 300

# $imageFolder = Join-Path $PSScriptRoot "CSV\Alfabet"
$imageFolder = Join-Path (Split-Path $openFileDialog.FileName -Parent) "Alfabet"

# Woorden ophalen en splitsen op spatie
$rawText = Get-Content $openFileDialog.FileName -Raw
$words = ($rawText -replace '\r','' -replace '\n',' ' -split '\s+') | Where-Object { $_ -ne '' }
$global:wordIndex = 0
$global:eersteKlik = $true $global:richtingVooruit
= $true

# GUI instellen
$form = New-Object System.Windows.Forms.Form
$form.Text = "Hebreeuwse Woordenweergave"
$form.Size = New-Object Drawing.Size(500,450)
$form.StartPosition = "CenterScreen"
$form.BackColor = "White"

# Panel voor letters + afbeeldingen
$panel = New-Object System.Windows.Forms.Panel
$panel.Location = New-Object Drawing.Point(0,50)
$panel.Size = New-Object Drawing.Size(500,450)
$panel.AutoSize = $false
$panel.AutoScroll = $true
$form.Controls.Add($panel)

$statusLabel = New-Object System.Windows.Forms.Label
$statusLabel.TextAlign = 'MiddleCenter'
$statusLabel.Size = New-Object Drawing.Size(230,30)
$statusLabel.Location = New-Object System.Drawing.Point(230, 10)
```

```
$statusLabel.Font = New-Object System.Drawing.Font("Segoe UI", 12,
[System.Drawing.FontStyle]::Regular)
$form.Controls.Add($statusLabel)

# Volgend woord knop
$nextBtn = New-Object System.Windows.Forms.Button
$nextBtn.Text = "Next word"
$nextBtn.Location = New-Object Drawing.Point(10,10)
$nextBtn.Size = New-Object Drawing.Size(100,30)
$form.Controls.Add($nextBtn)

# Event: knop gedrukt $nextBtn.Add_Click({
   if (-not $global:richtingVooruit) {
      $global:eersteKlik = $true
   }
   $global:richtingVooruit = $true

   $global:wordindex += 1
   $global:eersteKlik = $false

   # Doorspringen over onzichtbare/lege/koppelwoorden
   while ($global:wordIndex -lt $words.Length -and ($words[$global:wordIndex] -eq "" -or
$words[$global:wordIndex] -eq "-")) {
      $global:wordIndex++
   }

   if ($global:wordIndex -lt $words.Length) {
      ShowWord $words[$global:wordIndex]
      $prevBtn.Enabled = $true
   }
})

# Vorige woord knop
$prevBtn = New-Object System.Windows.Forms.Button
$prevBtn.Text = "←Prev. word"
$prevBtn.Location = New-Object Drawing.Point(120,10)
$prevBtn.Size = New-Object Drawing.Size(100,30)
$prevBtn.Enabled = $false $prevBtn.Add_Click({
   if ($global:richtingVooruit) {
      $global:eersteKlik = $true
   }
   $global:richtingVooruit = $false

   if ($global:eersteKlik) {
$global:wordindex -= 1
      $global:eersteKlik = $false
   } else {
      $global:wordindex -= 1
   }

   if ($global:wordindex -lt 0) {
      $global:wordindex = 0
```

```
    }

    while ($global:wordIndex -ge 0 -and ($words[$global:wordIndex] -eq "" -or $words[$global:wordIndex]
-eq "-")) {
        $global:wordIndex--         if
($global:wordIndex -lt 0) {
$global:wordIndex = 0
            break
        }
    }

    if ($global:wordIndex -le 0) {
        $prevBtn.Enabled = $false
    }

    ShowWord $words[$global:wordIndex]
})
$form.Controls.Add($prevBtn)

$longLabel = New-Object System.Windows.Forms.Label
$longLabel.Text = "Long Audio:"
$longLabel.Size = New-Object System.Drawing.Size(100, 20)
$longLabel.Location = New-Object System.Drawing.Point(120, 50)
$form.Controls.Add($longLabel)
$longLabel.BringToFront()

$longCheckBox = New-Object System.Windows.Forms.CheckBox
$longCheckBox.Name = "Long Audio"
$longCheckBox.Size =New-Object System.Drawing.Size(20, 20)
$longCheckBox.Location = New-Object System.Drawing.Point(220, 50)
$form.Controls.Add($longCheckBox)
$longCheckBox.BringToFront()

$playButton = New-Object System.Windows.Forms.Button
$playButton.Text = "> Play."
$playButton.Location = New-Object System.Drawing.Point(10, 50)
$playButton.Size = New-Object System.Drawing.Size(100, 30)
$playButton.Enabled = $true
$playButton.Add_Click({
    $audioBestanden = @()
    $currentWord = $words[$global:wordIndex]
$letters = $currentWord.ToCharArray()     foreach
($char in $letters) {
        $normalChar = NormalizeLetter $char
# $normalChar = NormalizeLetter $char
if ($longCheckBox.Checked) {
            $audioBestand = Join-Path $imageFolder "$normalChar-long.mp3"
       } else {
           $audioBestand = Join-Path $imageFolder "$normalChar.mp3"
        }
        if (Test-Path $audioBestand) {
$item = Get-Item $audioBestand
if ($item.Length -gt 0) {
```

```
            $audioBestanden += $audioBestand
          }
        }
    }
    SpeelAudioLijst $audioBestanden
})
$form.Controls.Add($playButton)
$playButton.BringToFront()

# ^^^^^ Copy before for AI Check  Checkblock 1 ^^^^^

function NormalizeLetter($char) {
    switch ($char) {
'ך' { return 'כ' }
        'ם' { return 'מ' }
        'ן' { return 'נ ' }
        'ף ' { return 'פ' }
'ץ' { return 'צ ' }
        default { return $char }
    }
}

function ShowWord($word) {
$panel.Controls.Clear()
    if (-not $word -or $word -eq "-") {
Write-Host "Skipped word: '$word'"        return
    }
    $letters = $word.ToCharArray()
    $lineHeight = $picSize + 10
    $maxLetters = 6
    $total = $letters.Length

    if ($total -le $maxLetters) {
        $statusLabel.Text = "Click pic to enlarge!"
        $statusLabel.ForeColor = [System.Drawing.Color]::White
        $statusLabel.BackColor = [System.Drawing.Color]::DarkBlue
    } else {
        $statusLabel.Text = "Scroll down with mouse!"
        $statusLabel.ForeColor = [System.Drawing.Color]::White
        $statusLabel.BackColor = [System.Drawing.Color]::DarkRed
    }

    # Dynamische hoogte berekening op basis van aantal letters
    $newHeight = [int]($lineHeight * $total)

    if ($total -le $maxLetters) {
        # 🔓 Korte woorden: autosize activeren
        $panel.AutoSize = $true
        $panel.AutoScroll = $false
        $panel.Size = New-Object Drawing.Size([int]($panel.Width),[int]( $newHeight+100))
$form.Size = New-Object Drawing.Size([int]($form.Width), [int]($newHeight+100))
    } else {
```

```
        # 🔓 Lange woorden: scroll activeren, autosize uit
        $panel.AutoSize = $false
        $panel.AutoScroll = $true
        $panel.Size = New-Object Drawing.Size([int]($panel.Width), [int]($maxLetters * $lineHeight))
        $form.Size = New-Object Drawing.Size([int]($form.Width), [int]($maxLetters * $lineHeight+100))
    }

    $startY = [Math]::Max(0, ($newHeight - ($total * $lineHeight)) / 2)

    for ($i = 0; $i -lt $total; $i++) {
        $char = $letters[$i]
        $normalChar = NormalizeLetter $char
        $yPos = [int]($startY + ($i * $lineHeight))

        # Zoek afbeelding (eerst PNG, dan JPG)
        $imgPathPNG = Join-Path $imageFolder "$normalChar.png"
        $imgPathJPG = Join-Path $imageFolder "$normalChar.jpg"

        $imgPath = $null

        if (Test-Path $imgPathPNG) {
$imgPath = $imgPathPNG
        } elseif (Test-Path $imgPathJPG) {
            $imgPath = $imgPathJPG
        }

        # Label met de letter blijft zoals hij is
        $lbl = New-Object System.Windows.Forms.Label
        $lbl.Text = $char
        $lbl.Font = New-Object Drawing.Font("Arial", $fntSize, [Drawing.FontStyle]::Bold)
        $lbl.Size = New-Object Drawing.Size([int]($picSize/2), $picSize)
        $lbl.TextAlign = 'MiddleCenter'
        $lbl.Location = New-Object Drawing.Point(50, $yPos)
        $lbl.Tag = $normalChar
        $panel.Controls.Add($lbl)

        # Afbeelding, tenzij het een koppelstreep is
    if ($char -ne "-" -and $imgPath) {

            $img = [System.Drawing.Image]::FromFile($imgPath)

            # Bereken bijbehorende breedte met behoud van aspect ratio
            $aspectRatio = $img.Width / $img.Height
            $calculatedWidth = [int]($picSize * $aspectRatio)

            # Stel PictureBox in
            $pic = New-Object System.Windows.Forms.PictureBox
            $pic.Size = New-Object Drawing.Size($calculatedWidth, $picSize)
            $pic.Location = New-Object Drawing.Point(250, $yPos)
            $pic.SizeMode = "Zoom"
            $pic.Image = $img
            $pic.Tag = $img
```

```
        $pic.Name = "$normalChar"

        $pic.Add_Click({
            Write-Host            $this.Name
$normalChar = $this.Name                  if
($longCheckBox.Checked) {
                $audioBestand = Join-Path $imageFolder "$normalChar-long.mp3"
            } else {
                $audioBestand = Join-Path $imageFolder "$normalChar.mp3"
            }

            SpeelAudioLijst($audioBestand)
        })

        try {
            $pic.Image = [System.Drawing.Image]::FromFile($imgPath)
        } catch {
continue
        }
        $panel.Controls.Add($pic)
    }
  }
}

function grootPic( $pic, $playTime){
            # Nieuw formulier maken
            $imgForm = New-Object System.Windows.Forms.Form
            $imgForm.Text = "Picture in true scale: $normalChar"
            $imgForm.StartPosition = "CenterScreen"
            $imgForm.BackColor = "White"

            # Haal afbeelding uit Tag
            $clickedImg =  $pic.Tag

            # Toon afbeelding op echte resolutie
            $fullBox = New-Object System.Windows.Forms.PictureBox
            $fullBox.Image =$clickedImg
            $fullBox.Size = $clickedImg.Size
            $fullBox.SizeMode = "Normal"
            $imgForm.ClientSize = $clickedImg.Size

            $imgForm.Controls.Add($fullBox)

            # Create a timer to close the form after $playTime seconds
            $timer = New-Object System.Windows.Forms.Timer
            $timer.Interval = $playTime
            $timer.Add_Tick({
                $timer.Stop()
                $imgForm.Close()
            })
            $timer.Start()

            $imgForm.ShowDialog()
```

```
}

function SpeelAudioLijst($audioBestanden) {     Add-
Type -AssemblyName presentationCore     foreach
($audio in $audioBestanden) {
      $ltr = [System.IO.Path]::GetFileNameWithoutExtension($audio)
      $ltr = $ltr -replace "-long",""

      $lbl = $panel.Controls | Where-Object {
         $_ -is [System.Windows.Forms.Label] -and $_.Tag -eq $ltr
      }
      if ($lbl) { $lbl.BackColor = 'Green' }
      [System.Windows.Forms.Application]::DoEvents()

      $player = New-Object System.Windows.Media.MediaPlayer
      $player.Open([Uri]::new($audio))
      while (-not $player.NaturalDuration.HasTimeSpan) {
         Start-Sleep -Milliseconds 333
      }
      $player.Play()

      if ($player.NaturalDuration.HasTimeSpan) {
         $duur = [int]$player.NaturalDuration.TimeSpan.TotalMilliseconds
         $pic = $panel.Controls | Where-Object {
            $_ -is [System.Windows.Forms.PictureBox] -and $_.Name -eq $ltr
         }
         if ($pic) {grootPic $pic $duur}
      } else {
         Start-Sleep -Milliseconds 333
      }
      $player.Stop()
      $player.Close()

      if ($lbl) { $lbl.BackColor = 'White' }
      [System.Windows.Forms.Application]::DoEvents()
      Start-Sleep -Milliseconds 333
   }
   [GC]::Collect()
   [GC]::WaitForPendingFinalizers()
}

# Toon eerste woord meteen
ShowWord $words[$global:wordIndex]

# Start GUI
$form.ShowDialog()
```

 # ^^^^^ Copy before for AI Check Checkblock 2 ^^^^^

Letterlijk.ps1: _____